อาจหาญ สัตยารักษ์

SOFT MAIL

การใช้เมนูภาษาไทย กราฟิก กับการรัน ไฟล์

เปิดเผยเคล็ดลับต่าง ๆ ในการใช้คำสั่ง Exec และการใช้เมนูแบบกราฟิก เพื่อเลือกรันไฟล์ตามต้องการ

ปัญหาที่มีผู้อ่านเขียนไปถามผู้เขียนมากที่สุดปัญหาหนึ่งคือการใช้คำสั่ง Exec ในเทอร์โบปาสคาล ซึ่งมีลักษณะคำถามที่เหมือน ๆ กัน คือ.

- สามารถรันไฟล์ *.BAT ได้หรือไม่
- สามารถป้อนพารามิเตอร์ต่อท้ายชื่อไฟล์ได้หรือไม่
- ขนาดหน่วยความจำที่จะต้องจองไว้นั้นควรมีขนาดเท่าใด
- สร้างเป็นเมนูแบบเลื่อนแถบสว่างโดยอยู่ในโหมดกราฟิกได้หรือไม่

วันนี้ผู้เขียนจะมาเปิดเผยเคล็ดลับต่าง ๆ ในการใช้คำสั่ง Exec และการใช้เมนูแบบกราฟิกเพื่อเลือกรันไฟล์ที่กำหนดได้ตามต้องการ และจะเป็นการตอบคำถามข้างต้นให้กระจ่างกันไปเลย

ในบทความฉบับนี้ค่อนข้างพิเศษกว่าเรื่องอื่น ๆ ที่ผ่านมา ทั้งนี้เพราะว่าผู้อ่านสามารถแยกโปรแกรมในฉบับนี้ออกไปเป็นงานย่อย ๆ ได้หลายอย่าง เช่น

1. โปรแกรมภาษาไทยกราฟิกโดยอาศัยฟอนต์อักขระจากไฟล์ font1.csl ของราชวิถีเวิร์ดพีซี ซึ่งมีเทคนิคพิเศษในการอ่านข้อมูลมาเพียงบางส่วนเท่านั้น พร้อมกับนำมาแสดงผลบนจอภาพ

2. นำไฟล์ font1.csl ที่ผู้เขียนได้ดัดแปลงข้อมูลให้เป็นอักขระที่สวยงามขึ้น ไปใช้แทนของเดิมในราชวิถีเวิร์ดพีซีได้ทันที

3. ได้วิธีการเข้าสู่กราฟิกโดยไม่ต้องอาศัยไฟล์ *.BGI อีกต่อไป

4. ได้ชุดคำสั่งที่ใช้ในการออกไปดอสชั่วคราวมาใช้งาน

5. ได้ชุดคำสั่งในการรันไฟล์โดยอาศัยคำสั่ง Exec ซึ่งจะสามารถรัน *.EXE, *.COM และ *.BAT

6. ได้กลุ่มคำสั่งด้านกราฟิกที่จำเป็นในการออกไปรันไฟล์ข้างนอกและสามารถกลับคืนมาที่กราฟิกเดิมโดยได้จัดทำเป็นซอฟต์เมล์เช่นเคย และมีไฟล์สำคัญ ๆ ดังนี้

1. ไฟล์ MAINMENU.PAS ไฟล์ทั้งหมดที่เสนอในบทความนี้

2. ไฟล์ HERCDRV.OBJ, V-GADRV.OBJ ซึ่งได้จากการเปลี่ยน herc.bgi และ egavga.bgi

3. ไฟล์ FONT1.CSL ที่ได้

เปลี่ยนแปลงข้อมูลใหม่ให้ได้ตัวอักขระที่สวยขึ้น

4. ไฟล์ MENU.TXT เป็นแฟ้มเอกสาร (รหัสเกษตร) ที่กำหนดรายการและชื่อไฟล์ของเมนูเลือก

ผู้เขียนเรียนขออนุญาตจาก น.พ. ชุษณะ มะกรสาร ในการใช้ไฟล์ font1.csl ของราชวิถีเวิร์ดพีซีมาสร้างอักขระกราฟิก และ อ.สุรศักดิ์ สงวนพงษ์ ในการตัดส่วนของฟังก์ชันจากหนังสือเทคนิคการเขียนโปรแกรมขั้นสูง แอดวานซ์เทอร์โบปาสคาล พิมพ์โดยบริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน) หน้า 154 มาพร้อมนี้ด้วย

ขอให้ผู้อ่านดูโปรแกรมในรูปที่ 1 ก่อน ซึ่งจะมีส่วนสำคัญ 2 ส่วนคือ การเข้ากราฟิกโหมดโดยไม่ต้องเรียกใช้ไฟล์ *.bgi และการอ่านข้อมูลจากไฟล์ font1.csl เพื่อนำมาสร้างอักขระแบบกราฟิก

ผู้เขียนขอตกลงตรงนี้อีกครั้งว่าสำหรับคำอธิบายการทำงานของโปรแกรมนั้น เป็นการตอบสนองผู้อ่านที่เพิ่งเริ่มเรียนภาษาปาสคาลเท่านั้น ถ้าท่านใช้ภาษาปาสคาลมานานแล้วควรละเว้นการอ่านคำอธิบายไป ทั้งนี้เพื่อจะได้เกิดการเรียนรู้ด้วยตนเอง เพราะผู้เขียนเชื่อว่าผู้อ่านนั้นมีหลายกลุ่ม ผู้อ่านที่เห็นว่าคำอธิบายเป็นส่วนเกินของบทความ ก็ให้นึกถึงน้อง ๆ ที่เพิ่งเริ่มเรียนภาษาปาสคาลด้วย

เข้ากราฟิกโดยไม่ใช้ไฟล์ *.BGI

เราสามารถนำไฟล์ *.BGI มารวมเข้ากับไฟล์ *.EXE ของเราได้ดังนี้

1. เปลี่ยนไฟล์ *.BGI ให้เป็นไฟล์ *.OBJ ก่อน โดยใช้ไฟล์ชื่อ BINOBJ.EXE ที่มีในแผ่นต้นฉบับของเทอร์โบปาสคาล ซึ่งมีรูปแบบการเรียกใช้ดังนี้

```
1: {$M 3000,0,4340}
2: uses crt,dos,graph;
3: var MenuST,Filename:array[1..16] of string;
4:     K,No,Max,Width,MaxL:byte;
5:     CommandFile,TimeST,Path,Owner:string;
6:     Data:array[0..4000] of byte;
7:     Font:array[0..4727] of byte;
8:     MonoScn:boolean;   Regs:registers;
9:     Grmode:integer;    Ch,Funckey:char;

10: PROCEDURE HercDriver; external;
11:   {$L Hercdrv.obj}

12: PROCEDURE EgaVgaDriver; external;
13:   {$L Vgadrv.obj}

14: PROCEDURE OpenGraph;
15: var GrDriver,Graphck :integer;
16: begin
17:   MonoSCN:=False;
18:   DetectGraph(GrDriver,GrMode);
19:   if GrDriver=7 then MonoSCN:=true else Grmode:=1;
20:   if MonoSCN then Graphck:=RegisterBgidriver(@HercDriver)
21:              else Graphck:=RegisterBgidriver(@EgavgaDriver);
22:   InitGraph(GrDriver, GrMode, '');
23: end;

24: PROCEDURE Readfont;
25: var f:file; k:word;
26:     FUNCTION SwapBit(B:Byte):byte;
27:      begin
28:       SwapBit:=B and 1 shl 7 + B and 2 shl 5  + B and 4 shl 3  +
29:                B and 8 shl 1 + B and 16 shr 1 + B and 32 shr 3 +
30:                B and 64 shr 5 + B and 128 shr 7;
31:      end;
32: begin
33:   assign(f,'font1.csl');
34:   reset(f,1);
35:   blockread(f,font,768);
36:   blockread(f,font,4727);
37:   close (f);
38:   if pred(Monoscn) then
39:     for k:=0 to 4727 do font[k]:=swapbit(Font[k]);
40: end;
```

รูปที่ 1
การเข้าสู่กราฟิกและการอ่านข้อมูลจากไฟล์ font1.csl ของราชวิถีเวิร์ดพีซี

binobj ชื่อไฟล์ *.bgi ชื่อไฟล์เมื่อเป็น *.obj ชื่อชุดคำสั่งเรียกในปาสคาล เช่น

```
A:\ >binobj herc.bgi hercdrv hercdriver
A:\ >binobj egavga.bgi vgadrv egavgadriver
```

เมื่อจบการทำงานเราจะได้ไฟล์ hercdrv.obj, vgadrv.obj ตามลำดับ

2. เขียนชุดคำสั่งเรียกใช้ไฟล์ *.obj ดังชุดคำสั่งในรูปที่ 1 ดังนี้

- บรรทัดที่ 10-11 เรียกกราฟิกสำหรับจอโมโนโครมธรรมดา

- บรรทัดที่ 12-13 เรียกกราฟิกสำหรับจอสีและจอวีจีเอโมโนโครม

3. เขียนชุดคำสั่งเข้ากราฟิกโหมดดังบรรทัดที่ 14-23 โดยกำหนดให้เป็นการเข้าสู่กราฟิกโหมดตามฮาร์ดแวร์ที่ตรวจพบ ให้สังเกตการเรียกใช้กราฟิกในบรรทัดที่ 20-21

ดังนั้นบรรทัดที่ 10-23 นี้ท่านสามารถนำไปใช้ในโปรแกรมกราฟิกอื่น ๆ ของท่านได้ โดยจะต้องสำเนาไฟล์ hercdrv.obj และ vgadrv.obj ไปใช้ด้วย

ในบรรทัดที่ 24-40 เป็นการอ่านข้อมูลจากไฟล์ font1.csl มาใช้งาน ขอให้ผู้อ่านสังเกตดังนี้

1. ไฟล์ font1.csl มีความยาว 6,144 ไบต์ เมื่อหารด้วย 256 จะได้ 24 ไบต์ แสดงว่าไฟล์นี้มีข้อมูล 256 ชุด ๆ ละ 24 ไบต์ ซึ่งคือรหัสแอสกี 0-255

2. ต้องการข้อมูลเท่าที่จำเป็นคือรหัสแอสกี 32-228 จึงมีความยาว (228-32+1)×24=4728 ไบต์ ส่วนรหัสแอสกี 229-255 นั้นเป็นสระผสมจะไม่อ่านเข้ามา โดยจะใช้เทคนิคการเขียนโปรแกรมทำงานแทน

3. ข้อมูลก่อนถึงรหัสแอสกี 32 (คือ 0-31) มีอยู่เท่ากับ 32×24=768 ไบต์

4. ดังนั้นเพื่อประหยัดหน่วยความจำเราจึงกำหนดมิติรองรับข้อมูลเพียง [0-4727] เท่านั้น

5. ผลจากการวิเคราะห์ข้างต้นจึงเขียนโปรแกรมอ่านข้อมูลดังนี้

- บรรทัดที่ 35 อ่านข้อมูลส่วนต้นทิ้งไป 768 ไบต์
- บรรทัดที่ 36 อ่านข้อมูลส่วนที่เป็นรหัสแอสกี 32-228 มา 4,727 ไบต์
- เนื่องจากเป็นการสร้างอักขระด้วยคำสั่ง setlinestyle จึงจำเป็นต้องสลับบิตของข้อมูลให้อยู่ในตำแหน่งที่ถูกต้องสำหรับการรันบนจอสีหรือจอวีจีเอโมโนโครม ด้วยคำสั่งบรรทัดที่ 38-39

ข้อมูลกลุ่มที่	ข้อมูลลำดับที่	รหัสแอสกี	การคำนวณหากลุ่มข้อมูล
0	0-23	32	32-32=0
1	24-47	33	33-32=1
2	48-71	34	34-32=2
	.	.	
	.	.	
195	4680-4703	227	227-32=195
196	4704-4727	228	228-32=196

ตารางที่ 1

```
41: PROCEDURE Showchar(x,y:integer; ASCII:byte);
42: var k:integer;
43: begin
44:  for k:=0 to 23 do
45:   begin
46:    if (font[(ASCII-32)*24+k] <> 0)  then
47:     begin
48:       if monoscn then setlinestyle(4,font[(ASCII-32)*24+k] shl 7,1)
49:           else setlinestyle(4,font[(ASCII-32)*24+k],1);
50:      Line(x,k+y,x+15,k+y);
51:     end;
52:   end;
53: end;

54: PROCEDURE Outthaixy(column,row:integer; content:string);
55: var k,ypix,xpix:integer;
56: begin
57:   column:=(column-1)*9;  row:=(row-1)*14;
58:   for k:=1 to length(content) do
59:    begin
60:      ypix:=0; xpix:=0;
61:      if content[k]=#207 then
62:        begin
63:          showchar(column-9,row,222); content[k]:=#206;
64:        end;
65:      if content[k] in [#217..#228] then xpix:=-9;
66:      if content[k] in [#215,#216] then xpix:=-8;
67:      if ((content[k])>#223) and ((content[k-1]>#216) or
68:                (content[k+1]=#207))  then ypix:=-3;
69:      if content[k] in [#32..#228] then
70:          showchar(column+xpix,row+ypix,ord(content[k]));
71:      if content[k] in [#32..#214] then  inc(column,9);
72:    end;
73:    setlinestyle(0,255,1);
74: end;
```

รูปที่ 2 การแสดงภาษาไทยกราฟิกโดยอาศัยข้อมูลจากไฟล์ font1.csl

สร้างภาษาไทยกราฟิกจาก font1.csl

จากการอ่านข้อมูลตามที่ได้กล่าวมาแล้วนั้น ข้อมูลจะแบ่งออกเป็นกลุ่ม ๆ ละ 24 ไบต์ ซึ่งถ้าพิจารณาจะได้ข้อมูลมีลักษณะดังตารางที่ 1

นั่นคือ ถ้าให้ ASCII คือ ตัวแปรแทนรหัสแอสกี จะหาข้อมูลได้ดังนี้

1. หากลุ่มข้อมูลได้จาก ASCII-32

2. ข้อมูลของอักขระนั้นจึงเริ่มที่ข้อมูลที่ (ASCII-32)*24 ไปจนถึง (ASCII-32)*24+23

ตัวอย่างเช่น

รหัสแอสกี 34 ข้อมูลกลุ่มที่ 34-32=2 ข้อมูลจึงเป็นลำดับที่ 2*24=48 ถึง 48+23=71

รหัสแอสกี 65 ข้อมูลกลุ่มที่ 65-32=33 ข้อมูลจึงเป็นลำดับที่ 33*24=792 ถึง 792+23=815

เมื่อเขียนชุดคำสั่งแสดงอักขระกราฟิกบนจอภาพด้วยคำสั่ง setlinestyle และ line จะได้เป็นดังโปรแกรมในรูปที่ 2 ซึ่งเป็นชุดคำสั่งที่มีความยุ่งยากน้อยมาก เพราะไม่มีการกำหนดสีให้ข้อความในชุดคำสั่ง โดยเมื่อต้องการ

ข้อความเป็นสีใด ให้กำหนดด้วยคำสั่ง setcolor เองก่อนที่จะแสดงข้อความกราฟิก ทั้งนี้เกิดจากเหตุผลที่เราต้องการให้โปรแกรมใช้หน่วยความจำน้อยที่สุดเท่าที่จะทำได้

เนื่องจากชุดคำสั่งในรูปที่ 2 นี้เป็นเพียงส่วนประกอบของโปรแกรมนี้เท่านั้น ผู้เขียนจึงขอให้ผู้อ่านทำความเข้าใจกับคำสั่งต่าง ๆ เอาเอง เพราะว่าคงชินกับชุดคำสั่งแบบนี้ที่ผู้เขียนใช้ในการแสดงภาษาไทยมาหลายครั้งแล้ว

การกำหนดขอบเขตข้อความ

คราวนี้มาดูแนวคิดของการเขียน "เมนูแบบเลื่อนแถบสว่างที่สามารถรันไฟล์" นี้ก่อน ซึ่งผู้เขียนใช้แนวคิดดังนี้

1. รายการในเมนูหลักและชื่อไฟล์ที่จะรันนั้น กำหนดให้เป็นแฟ้มเอกสารภายนอกในชื่อ menu.txt

2. จำนวนรายการของเมนูทั้งหมดจะมีระหว่าง 2-16 รายการ

3. แต่ละรายการในเมนูแบบเลื่อนแถบสว่างนั้นจะมีได้ไม่เกิน 24 คอลัมน์ ถ้าป้อนไว้เกินจะตัดส่วนที่อยู่ตั้งแต่คอลัมน์ที่ 25 ออกไป (หมายถึงคอลัมน์ในแฟ้มเอกสารที่เราป้อนไฟล์ menu.txt)

4. ชื่อไฟล์ที่จะระบุให้รันในแต่ละรายการนั้น ให้ป้อนต่อท้ายรายการนั้น ๆ โดยจะต้องระบุไดรฟ์ พาธ และชื่อไฟล์ให้ครบถ้วน (ต้องใช้สกุลของไฟล์ด้วย) โดยใช้เครื่องหมาย [ ] เป็นตัวกำหนดขอบเขตของชื่อไฟล์ที่จะรัน (ในบทความต่อจากนี้ไปคำว่า ชื่อไฟล์ จะหมายถึงพาธและชื่อไฟล์รวมกัน)

ตัวอย่างของไฟล์ menu.txt ที่มีในซอฟต์เมล์จะเป็นดังนี้

โปรแกรมพิมพ์เอกสาร RW
[C:\RW\RW.EXE]
โปรแกรมพิมพ์เอกสาร CW
[C:\DOS\CW.EXE]
ทำคะแนนสอบปลายภาค
[C:\FINAL\FINAL.EXE]
วิเคราะห์ข้อสอบ-ตรวจข้อสอบ
[C:\ITEM\ITEM.EXE]
โปรแกรม THAISHOW 3.0
[C:\TS30\THAISHOW.EXE SAMPLE]
โปรแกรมฐานข้อมูล dBASE
[C:\DBASE\DBASE4.BAT]
โปรแกรมโลตัส 1-2-3
[C:\LOTUS\lotus.bat]
"พิมพ์บัตร" เอนกประสงค์
[C:\ITEM\ITEM.EXE]
โปรแกรม THAI BANNER
[C:\BANNER\TBANNER.EXE]
โปรแกรมพิมพ์ซองจดหมาย
[A:postpnt.exe]
เทอร์โบปาสคาล
[C:\TURBO\TURBO.EXE]

โปรดสังเกตว่าส่วนท้ายสุดของทุกรายการจะกำหนดชื่อไฟล์และพาธที่ต้องการ จะใช้ตัวเล็กหรือตัวใหญ่ก็มีผลเท่ากัน เพราะโปรแกรมจะอ่านให้เป็นตัวใหญ่ทั้งหมด ทั้งสามารถกำหนดให้รันไฟล์ได้ทั้ง *.EXE, *.COM และ *.BAT รวมทั้งสามารถกำหนดพารามิเตอร์ต่อท้ายไฟล์ที่รันได้ด้วย

เนื่องจากว่าแนวคิดของการป้อนไฟล์ menu.txt มีการกำหนดขอบเขตของรายการด้วย เราจึงต้องเขียนโปรแกรมตรวจสอบ ซึ่งจะได้โปรแกรมในรูปที่ 3 โดยมีรายละเอียดดังนี้

```
75: PROCEDURE ClsGR(x1,y1,x2,y2:integer);
76: var ViewPort:ViewPortType;
77: begin
78:    GetViewSettings(ViewPort);
79:    setviewport(x1,y1,x2,y2,clipon);  clearviewport;
80:    setviewport(ViewPort.x1,ViewPort.y1,ViewPort.x2,
81:                ViewPort.y2,ViewPort.Clip);
82: end;

83: PROCEDURE ReadChar;
84: begin
85:   Regs.AH:=$0;  intr($16,regs);
86:   Funckey:=#0;  ch:=chr(regs.AL);
87:   if ch=#0 then Funckey:=chr(regs.AH);
88: end;

89: FUNCTION CheckSTR(Content:string):string;
90: var k,LL:byte;  ST:string;
91: begin
92:    if Content[length(Content)]=#32 then
93:    begin
94:       repeat
95:         delete(content,length(Content),1);
96:       until (Content[length(Content)]<>#32) or (Content='');
97:   end;
98:   LL:=0;  ST:='';
99:   if Content<>'' then
100:   begin
101:     for k:=1 to length(Content) do
102:      begin
103:       if Content[k] in [#32..#214] then inc(LL);
104:       if LL<=24 then ST:=ST+Content[k];
105:     end;
106:     If Width<LL then Width:=LL;
107:   end;
108:   CheckSTR:=ST;
109: end;
```

รูปที่ 3
การลบจอภาพ ตรวจการกดแป้น และการตรวจขอบเขตของข้อความ

1. บรรทัดที่ 75-82 เป็นคำสั่งลบจอภาพในขอบเขตที่กำหนด ซึ่งถ้าดูเผิน ๆ จะคล้ายกับที่เคยเขียนมาหลายครั้งแล้ว แต่ที่จริงคราวนี้มีลักษณะพิเศษกว่า คือ

- บรรทัดที่ 78 เก็บค่าต่าง ๆ ในการกำหนดขอบเขตจอภาพที่มีก่อนหน้านี้ ไว้ในตัวแปร Viewport

- บรรทัดที่ 79 ลบจอภาพในขอบเขตที่กำหนด

- บรรทัดที่ 80-81 คืนค่าต่างของการกำหนดขอบเขตจอภาพจาก Viewport คืนกลับ

2. ฟังก์ชัน CheckSTR ซึ่งทำหน้าที่ในการตัดข้อความให้อยู่ในขอบเขตที่กำหนด มีดังนี้

- บรรทัดที่ 92-97 ตัดรหัสแอสกี 32 (ช่องว่างในข้อความ) ที่อยู่ส่วนท้ายของข้อความออกไป โดยจะวนรอบตัดรหัสแอสกีท้ายข้อความออกไปเรื่อย ๆ จนกว่าจะพบรหัสแอสกีอื่น หรือข้อความนั้นว่างลง

- บรรทัดที่ 98-107 เป็นการตัดส่วนข้อความส่วนที่เกินคอลัมน์ที่ 24 ออกไป โดยอาศัยตัวแปร LL มาช่วยกำหนด โอนข้อความจากตัวแปร content ไปไว้ในตัวแปร ST ในกรณีที่ค่า LL มีค่าไม่เกิน 24 ซึ่งตามแนวคิดนี้เราจะยังได้สระที่อยู่ในคอลัมน์ที่ 24 ด้วยทุกครั้ง

ค่า LL จะเพิ่มขึ้นเฉพาะกรณีที่รหัสแอสกีเป็นอักขระปกติเท่านั้น (32-214)

- บรรทัดที่ 106 ถ้าตัวแปร width <LL ก็ให้ค่าตัวแปร width เท่ากับ LL ตัวแปร width นี้จะเป็นตัวกำหนดความกว้างของเมนูเลื่อนแถบสว่าง เพื่อให้ปรับความกว้างได้ตามกลุ่มรายการที่กำหนด

```
110: PROCEDURE ReadText;
111: var f:file; Size,K:word;  P,PP:byte;
112: begin
113:    Max:=99;     assign(f,'Menu.txt');
114:   {$I-} reset(f,1);
115:    if IOResult=0 then
116:     begin
117:       size:=filesize(f);
118:       if size>4000 then Size:=4000;
119:       blockread(f,data,size);
120:       close(f);
121:       Width:=0;  K:=0;  No:=1; MenuST[No]:='';
122:       repeat
123:         if  Data[k] in [32..228] then
124:             MenuST[No]:=MenuST[No]+Upcase(chr(Data[k]));
125:         if Data[k] in [10,26] then
126:          begin
127:           if MenuST[No]<>'' then
128:            begin
129:              if No in [1..16] then
130:              begin
131:                P:=pos('[',MenuST[no]);  PP:=pos(']',MenuST[no]);
132:                Filename[no]:='';
133:                if (P<>0) and ((PP<>0) and (PP>P)) then
134:                 begin
135:                   FileName[No]:=Copy(MenuST[No],P+1,PP-P-1);
136:                   MenuST[No]:=CheckSTR(Copy(MenuST[No],1,P-1));
137:                 end;
138:              end;
139:              Max:=No; Inc(NO);
140:              if No in [1..16] then MenuST[No]:='';
141:            end;
142:           if (No=17) or (Data[k]=26) then k:=size;
143:          end;
144:         inc(k);
145:       until k>size;
146:     end;
147: end;
```

รูปที่ 4
การอ่านข้อความจากแฟ้มเอกสาร menu.txt แยกเป็นข้อความและชื่อไฟล์

อ่านรายการเมนูจากแฟ้มเอกสาร

จากแฟ้มเอกสาร menu.txt ตัวอย่างที่ผ่านมา เมื่อเราเขียนโปรแกรมในการอ่านแฟ้มเอกสารนั้นจะต้องทำงานได้ดังนี้

1. แยกรายการของเมนูและชื่อไฟล์ออกจากกันได้

2. ตรวจขอบเขตของรายการเมนูให้มีได้ไม่เกิน 24 คอลัมน์

3. กำหนดให้อ่านได้ไม่เกิน 16 รายการ

4. ถ้าหาแฟ้มเอกสาร menu.txt ไม่พบ โปรแกรมจะไม่ทำงาน

เมื่อนำมาเขียนโปรแกรมจะได้ดังรูปที่ 4 โดยผู้เขียนใช้ตัวแปร MenuST[1]-MenuST[16] มารองรับรายการของเมนู และ Filename[1]-FileName[16] มารองรับชื่อไฟล์ที่จะรัน

รายละเอียดของโปรแกรมในรูปที่ 4 มีดังนี้

1. กำหนดตัวแปร Max=99 เพื่อใช้เป็นเงื่อนไขว่าหาแฟ้มเอกสาร Menu.txt พบหรือไม่ ทั้งยังใช้กำหนดจำนวนสูงสุดของรายการเมนูที่อ่านได้

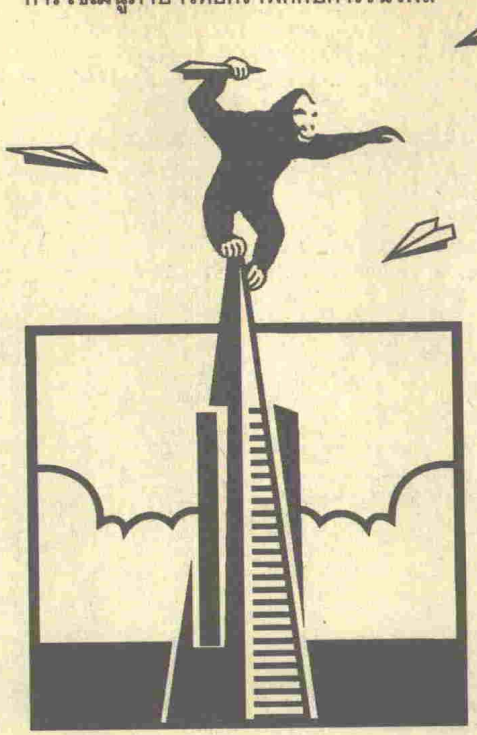
2. บรรทัดที่ 113-114 ตรวจสอบแฟ้มเอกสาร menu.txt ว่ามีอยู่หรือไม่

3. บรรทัดที่ 115-146 ส่วนการอ่านรายการในเมนูและชื่อไฟล์

4. บรรทัดที่ 117-120 หาความยาวแฟ้มเอกสาร อ่านข้อมูลเข้าตัวแปร data

5. บรรทัดที่ 121 กำหนดค่าให้ตัวแปรเพื่อเริ่มต้นทำงาน

6. บรรทัดที่ 123-124 ถ้าข้อมูล



เป็นรหัสแอสกี 32-228 ให้เก็บเข้าตัวแปร MenuST[No] โดยจะเปลี่ยนอักขระตัวเล็กให้เป็นตัวใหญ่

7. บรรทัดที่ 125-143 เมื่อพบรหัสเลื่อนบรรทัด (#10) หรือรหัสจบไฟล์ (#26] จะทำการประมวลผลเพื่อแยกชื่อไฟล์และรายการในเมนูออกจากกัน โดยการตรวจสอบจาก [และ] ที่ป้อนคร่อมชื่อไฟล์ไว้ ทำให้ได้ Filename [no] แยกออกมา ส่วนตัวแปร MenuST [no] จะถูกตรวจสอบขอบเขตไม่ให้เกิน 24 คอลัมน์

8. บรรทัดที่ 139 กำหนดตัวแปร Max เพื่อเก็บค่าสูงสุดของรายการในเมนู

9. บรรทัดที่ 142 ถ้ารายการในเมนูมากกว่า 16 หรือหมดข้อความที่เขียนไว้ก็จะจบการทำงาน

การใช้คำสั่ง EXEC

จุดประสงค์หลักของบทความนี้อยู่ที่การออกไปรันไฟล์ข้างนอก โดยรันไฟล์ตามรายการที่กำหนดในเมนูจำนวนสูงสุด 16 รายการ แต่เพื่อความสมบูรณ์ของโปรแกรม ผู้เขียนจึงได้กำหนดทางเลือกไว้ให้อีกทางหนึ่งคือการออกไปยังดอสชั่วคราว ซึ่งจะมีผลให้ออกไปดำเนินการเกี่ยวกับคำสั่งดอสได้ทั้งหมด

ในกรณีรันไฟล์ข้างนอกหรือการออกไปดอสชั่วคราวนี้จะต้องอาศัยคำสั่ง Exec เข้ามาช่วยงาน ซึ่งในเทอร์โบ-ปาสคาล (หนังสือ TURBO PASCAL Owner's Handbook หน้า 403-404) ได้กำหนดรูปแบบไว้ดังนี้

```
Exec(Path,CmdLine:string);
```

เมื่อ Path คือชื่อไฟล์ที่จะรัน และ CmdLine คือพารามิเตอร์ที่จะกำหนดตามหลังชื่อไฟล์ ตัวอย่างเช่น

1. เมื่อรันผ่านดอส C:\WORD >rwmain เขียนคำสั่งเป็น

```
Exec('C:\WORD\RWMAIN.EXE','');
```

2. เมื่อรันผ่านดอส C:\TS30> thaishow sample เขียนคำสั่งเป็น

```
Exec('C:\TS30\THAISHOW.EXE','
SAMPLE');
```

3. เมื่อรันผ่านดอส C:\TS30> thaishow sample logo เขียนคำสั่งเป็น

```
Exec('C:\TS30\THAISHOW.EXE','
SAMPLE LOGO');
```

นั่นคือสามารถส่งพารามิเตอร์ตามหลังไฟล์ได้แบบเดียวกับที่รันผ่านดอสตามปกติ โดยในหนังสือได้ระบุว่าถ้าเป็นการใช้งานโดยอาศัยไฟล์ command.com จะต้องกำหนดรูปแบบดังนี้

```
Exec('\COMMAND.COM','/C DIR *.PAS');
```

การรันไฟล์ *.BAT นั้นเป็นการรันโดยอาศัยไฟล์ command.com ด้วย ดังนั้นการรันจึงไม่สามารถใช้คำสั่งแบบกลุ่ม *.COM, *.EXE ได้ นั่นคือคำสั่ง Exec('C:\LOTUS\LOTUS.BAT','');) จะไม่สามารถรันไฟล์ lotus.bat ได้

ในกรณีนี้เราจะต้องใช้คำสั่งว่า

```
Exec('\COMMAND.COM','/C LOTUS.BAT');
```

สำหรับตำแหน่งที่อยู่ของไฟล์ command.com นั้นเราสามารถใช้คำสั่งตรวจหาได้ว่าไฟล์นี้อยู่ในพาธใดได้ ถ้าท่านใช้งานกับแผ่นดิสก์จะต้องระวังเรื่องไฟล์ command.com ให้มาก เพราะไฟล์นี้จะต้องเป็นรุ่นเดียวกันกับไฟล์ที่เรียกเข้ามาในตอนเปิดเครื่อง เช่น ท่านเปิดเครื่อง (boot) ด้วยดอส 5.0 แผ่นที่จะนำมาใช้รวมกับโปรแกรมนี้จะต้องมีไฟล์ command.com เป็นรุ่น 5.0 ด้วย คำสั่ง Exec (กรณีที่รัน *.BAT) จึงจะทำงานได้

ทั้งนี้จะต้องกำหนดหน่วยความจำสแต็ก (stack) และหน่วยความจำฮีฟ (heap) ให้เหมาะสมด้วย

การออกไปดอสชั่วคราว

การออกไปดอสชั่วคราวนั้น ตามหลักการจะต้องเป็นไปดังนี้

1. ยกเลิกการทำงานของโปรแกรมที่กำลังรันอยู่เป็นการชั่วคราวก่อน

2. ใช้คำสั่ง Exec ในกรณีที่ต้องอาศัยไฟล์ command.com เข้ามาทำงาน โดยเราจะต้องระบุไฟล์ command.com ให้ตรงกับตำแหน่งที่อยู่ของไฟล์นี้ และจะต้องเป็นดอสรุ่นเดียวกันกับที่กำลังใช้งานด้วย

3. การใช้คำสั่ง Exec รันไฟล์ command.com โดยไม่ระบุพารามิเตอร์จะช่วยให้เราทำงานได้เช่นเดียวกันกับอยู่ในดอสปกติ และได้ทุกคำสั่ง

4. เมื่อต้องการกลับคืนไปรันโปรแกรมเดิมก่อนการออกมาดอสชั่วคราว ให้ป้อนคำว่า exit

แต่ตามแนวคิดที่ผู้เขียนเสนอในบทความ จะปรับปรุงให้แสดงข้อความ

```
148: FUNCTION GetEnViron:string;
149: type Environment=array[1..32767] of char;
150: var Environ:^Environment;  Environseg:word;
151:     st,Env:string;         I:byte;
152: begin
153:   Environseg:=(mem[Prefixseg:$002C]) or (mem[Prefixseg:$002D] shl 8);
154:   GetEnviron:='';    Env:='COMSPEC'+'=';
155:   Environ:=ptr(Environseg,0);  I:=1;
156:   while Environ^[i]<>#00 do
157:    begin
158:      st:='';
159:      repeat
160:        st:=st+Environ^[i]; inc(I);
161:      until Environ^[i]=#00;
162:       if (length(st) >= length(Env))  and
163:          (copy(st,1,length(Env)) = Env) then
164:             GetEnviron:=copy(st,length(Env)+1,length(ST)-length(Env))
165:        else Inc(i);
166:    end;
167: end;

168: PROCEDURE ExecDos;
169: var ComLine,PathActive:string;
170: begin
171:   CommandFile:=GetEnViron; clrscr;
172:   repeat
173:      getdir(0,PathActive);  PathActive:=PathActive+'>';
174:      writeln;   textcolor(0);  TextbackGround(7);
175:      if Pred(Monoscn) then
176:       begin
177:         textcolor(15);  TextbackGround(9);
178:       end;
179:      write(' Type EXIT to return to MAINMENU... ');
180:      textcolor(7);  TextbackGround(0);
181:      writeln(#13#10);  write(PathActive);
182:      Readln(ComLine);
183:      for k:=1 to length(ComLine) do ComLine[k]:=upcase(ComLine[k]);
184:      if ComLine<>'EXIT' then Exec(CommandFile,'/C '+ComLine);
185:  until ComLine='EXIT';
186: end;
```

รูปที่ 5 ส่วนการหาไฟล์ command.com และชุดคำสั่งออกไปดอสชั่วคราว

ระบุให้ป้อนคำว่า exit หลังการทำงานของทุกคำสั่งเสมอ ทั้งนี้เพื่อเป็นการเตือนให้รู้ว่าขณะนี้เราออกมาดอสชั่วคราวเท่านั้น ไม่ใช่อยู่ในดอสจริง ๆ ซึ่งจะต่างจากการออกมาดอสชั่วคราวที่ผู้อ่านเคยเห็นในโปรแกรมอื่น ๆ ที่ออกมานาน ๆ ผู้ใช้จะลืมว่ากำลังรันโปรแกรมอื่นค้างไว้อยู่ นำมาเขียนโปรแกรมดังรูปที่ 5

สำหรับบรรทัดที่ 148-167 นั้นมาจากหนังสือ เทคนิคการเขียนโปรแกรมขั้นสูง แอดวานซ์เทอร์โบปาสคาล ของ อ.สุรศักดิ์ สงวนพงษ์ พิมพ์โดยบริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน) หน้า 154 ดังที่แจ้งไว้ข้างต้นแล้ว จะเป็นการค้นหาเส้นทางของไฟล์ command.com จากสภาวะแวดล้อมว่ามาจากพาธใด ซึ่งจะช่วยให้ค้นพาไฟล์นี้พบแน่นอน

รายละเอียดอื่น ๆ มีดังนี้

1. บรรทัดที่ 171 หาไฟล์ command.com

2. บรรทัดที่ 172-185 ส่วนการวนรอบใช้คำสั่ง Exec ซึ่งจะทำให้ออกไปดอสชั่วคราว

3. บรรทัดที่ 173 หาพาธที่กำลังติดต่ออยู่ เพื่อนำไปแสดงในบรรทัดที่ 181

4. บรรทัดที่ 174-180 แสดงข้อความในบรรทัดที่ 179 สีขาวบนพื้นสีน้ำเงินสำหรับจอสี และข้อความสีดำบนพื้นขาวในจอโมโนโครมธรรมดา

5. บรรทัดที่ 182-183 อ่านคำสั่งที่จะใช้งานผ่านดอส โดยจะเปลี่ยนข้อความที่อ่านได้ให้เป็นตัวใหญ่ทั้งหมด

6. บรรทัดที่ 184 ถ้าข้อความที่ป้อนไม่ใช่ EXIT ก็จะเรียกใช้ command.com มาทำงาน

โปรแกรมในรูปที่ 5 นี้ ผู้อ่านสามารถนำไปใช้ในโปรแกรมอื่น ๆ ได้

การรันไฟล์แบบกำหนดพารามิเตอร์

การทำงานตามชุดคำสั่ง ExecDos นั้นเป็นการออกไปดอสชั่วคราว ซึ่งไม่ตรงกับจุดประสงค์ในการสร้างเมนูแบบเลื่อนแถบสว่างเพื่อรันไฟล์ เราจึงต้องเขียนชุดคำสั่งในการรันไฟล์ผ่านโปรแกรมอีกหนึ่งชุดคำสั่ง โดยจะต้องมีความสามารถดังนี้

1. ต้องกลับมารันไฟล์กราฟิกเดิมก่อนออกไปได้

2. ต้องรันไฟล์ *.COM, *.EXE, *.BAT ได้ทุกไฟล์

3. ต้องป้อนพารามิเตอร์ต่อท้ายไฟล์ *.COM, *.EXE ได้

4. เพื่อความถูกต้องของการรันไฟล์ที่ระบุ เพราะว่าไฟล์บางไฟล์อาจเรียกใช้ไฟล์อื่นจากแผ่นดิสก์เข้ามาใช้งานด้วย เช่น อ่านข้อมูลสำหรับอักขระในการพิมพ์ อ่านแฟ้มภาพ ฯลฯ ต้องให้มีการเปลี่ยนพาธไปยังส่วนของโปรแกรมนั้น ๆ ก่อนเสมอ

นำแนวคิดดังกล่าวมาเขียนโปรแกรมได้ดังรูปที่ 6 โดยมีรายละเอียดดังนี้

1. บรรทัดที่ 192-196 ตรวจแยกพาธและชื่อไฟล์ออกจากกัน ทั้งนี้เพราะว่าส่วนต้นของข้อความเป็นพาธ และส่วนท้ายของข้อความจะเป็นชื่อไฟล์และพารามิเตอร์ โดยตรวจจากอักขระ \ และ :

```
187: PROCEDURE RunFile(STT:string);
188: var k:byte; Filename,NewPath:string;
189: begin
190:  if STT[2]=':' then
191:   begin
192:      k:=length(STT);  Filename:='';
193:      repeat
194:        Filename:=STT[k]+Filename;
195:        dec(k);
196:      until (STT[k] in ['\',':']);
197:      if STT[k]=':' then NewPath:=Copy(STT,1,k)
198:                    else NewPath:=Copy(STT,1,k-1);
199:      if (NewPath[3]='\') and (NewPath[0]=#3) then NewPath[0]:=#2;
200:     {$I-} chdir(NewPath);
201:      if IOResult<>0 then Chdir(Path);
202:     {$I+}
203:    k:=pos('.',Filename);
204:    if k<>0 then
205:      begin
206:        STT:=Copy(Filename,k+1,3);
207:        if STT='BAT' then
208:         begin
209:          CommandFile:=GetEnViron;
210:          Exec(CommandFile,'/C '+Filename);
211:         end
212:          else
213:           begin
214:            STT:='';
215:            if Length(FileName)>k+3 then
216:             begin
217:               STT:=Filename; delete(STT,1,K+3);
218:               FileName:=Copy(Filename,1,k+3);
219:             end;
220:            Exec(Filename,STT);
221:           end;
222:      end;
223:   end;
224: end;
```

รูปที่ 6
การเปลี่ยนพาธและรันไฟล์ที่กำหนดไว้ในแฟ้มเอกสาร menu.txt

2. บรรทัดที่ 197-199 จะได้พาธของไฟล์เก็บในตัวแปร NewPath

3. บรรทัดที่ 200-202 ส่วนการเปลี่ยนพาธที่ตรวจสอบความผิดพลาดด้วย

4. บรรทัดที่ 206-211 ส่วนการรันไฟล์ *.BAT ซึ่งจะต้องอาศัยไฟล์ command.com

5. บรรทัดที่ 214-219 แยกชื่อไฟล์และพารามิเตอร์ออกจากกัน

6. บรรทัดที่ 220 รันไฟล์พร้อมกับพารามิเตอร์ที่ตรวจพบ

ส่วนของโปรแกรมหลัก

คราวนี้มาถึงส่วนสำคัญคือ การแสดงรายการในเมนูและการเลือกรายการในเมนู โดยผู้เขียนได้กำหนดไว้ดังนี้

1. ถ้ารายการเมนูมีไม่ถึง 2 รายการจะไม่ทำงาน

2. ถ้ารายการเมนูมีระหว่าง 2-8 รายการจะแสดงเพียงคอลัมน์เดียว ถ้ามีมากกว่า 8 รายการจะแบ่งแสดงเป็น 2 คอลัมน์

3. รายการที่แสดงบนจอนั้นจะปรับตำแหน่งให้เหมาะสมกับกรอบภาพโดยอัตโนมัติ

4. การใช้แป้นลูกศรเพื่อเลือกรายการในเมนู จะตรวจสอบการเลื่อนให้เป็นไปตามเงื่อนไขดังนี้

4.1 ถ้าใช้แป้นลูกศรเลื่อนลง เมื่อเลื่อนลงไปถึงรายการสุดท้าย จะกลับมาที่รายการแรก ยกเว้นกรณีที่มีรายการเป็น 2 คอลัมน์จะเลื่อนไปยังรายการแรกของคอลัมน์อื่น

4.2 ถ้าใช้แป้นลูกศรเลื่อนขึ้น เมื่อเลื่อนขึ้นไปถึงรายการแรก จะกลับมาที่รายการสุดท้าย ยกเว้นกรณีที่มีรายการเป็น 2 คอลัมน์จะเลื่อนไปยังรายการสุดท้ายของคอลัมน์อื่น

4.3 ถ้าใช้แป้นลูกศรซ้ายหรือขวา จะเลื่อนกลับไปมาในแนวนอน (เฉพาะกรณีที่มี 2 คอลัมน์)

5. ในกรณีการออกไปรันไฟล์ข้างนอก จะต้องทำดังนี้

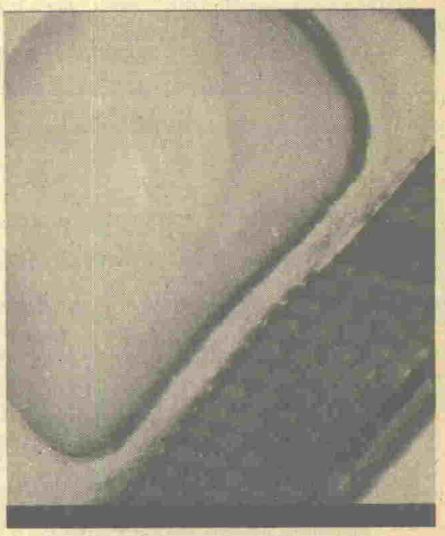
5.1 ต้องจำกราฟิกใช้งานอยู่ขณะนั้นไว้ก่อนด้วยคำสั่ง RestoreCrtmode;

5.2 ออกไปรันไฟล์ข้างนอก

5.3 กลับเข้าสู่กราฟิกใช้งานใหม่ด้วยคำสั่ง Setgraphmode(Grmode); ทั้งนี้เพราะตอนที่เราเข้าสู่กราฟิกนั้น เราใช้ตัวแปร Grmode เป็นตัวกำหนดกราฟิกใช้งาน

นำมาเขียนโปรแกรมหลักได้ดังรูปที่ 7 มีรายละเอียดดังนี้

1. บรรทัดที่ 226 กำหนดตัวแปรเพื่อแยกรายการของเมนูออกตามจำนวนที่อ่านได้ เช่น ถ้ามี 9 รายการจะแบ่งออกเป็น 5 และ 9-5=4 ถ้ามี 10 รายการก็จะแบ่งเป็น 5 และ 10-5=5



```
225: PROCEDURE Mainmenu;
226: Const MaxC:array[1..16] of byte=(1,2,3,4,5,6,7,8,5,5,6,6,7,7,8,8);
227: var choice,DisPlay,No:byte;
228:     Xleft,Yup,XColumn1,XColumn2:word;

229:       Procedure Menu;
230:        begin
231:          MaxL:=MaxC[Max];   YUp:=50+(8-MaxL)*14;
232:          setcolor(3);  setfillstyle(9,getcolor);
233:          bar(0,0,639,346);  setlinestyle(0,$FFFF,3);
234:          setcolor(12); rectangle(1,1,638,346);
235:          clsgr(60,8,579,38);  setcolor(15); rectangle(60,8,579,38);
236:          No:=0;
237:          for k:=1 to length(Owner) do
238:                 if Owner[k] in [#32..#214] then Inc(no);
239:          setcolor(11); Outthaixy(37-no div 2,2,Owner);
240:          if Max<=8 then Xleft:=(36-(Width div 2))
241:                    else  Xleft:=(20-(Width div 2));
242:          clsgr(Xleft*9-9,Yup,(Xleft+Width)*9+3,Yup+3+MaxL*28);
243:          setcolor(10); setlinestyle(0,$FFFF,3);
244:          rectangle(Xleft*9-9,Yup-1,(Xleft+Width)*9+3,Yup+3+MaxL*28);
245:          setcolor(14);
246:          for k:=1 to maxL do
247:                Outthaixy(Xleft+1,k*2+3+(8-MaxL),Menust[k]);
248:          XColumn1:=Xleft;
249:          if max>8 then
250:           begin
251:            XColumn1:=Xleft;  XColumn2:=(52-(Width div 2));
252:            clsgr(XColumn2*9-9,Yup,(XColumn2+Width)*9+3,Yup+3+MaxL*28);
253:            setcolor(10); setlinestyle(0,$FFFF,3);
254:            rectangle(XColumn2*9-9,Yup-1,(XColumn2+Width)*9+3,Yup+3+MaxL*28);
255:            setcolor(14);
256:            for k:=MaxL+1 to Max do
257:               Outthaixy(XColumn2+1,(k-MaxL)*2+3+(8-MaxL),Menust[k]);
258:           end;
259:            setcolor(11); setfillstyle(1,getcolor);
260:            bar(500,294,613,326);
261:            setcolor(14); rectangle(500,293,613,326);
262:            setcolor(0);  Outthaixy(57,22,'Esc เลิกงาน');
263:            Outthaixy(57,23,'F10 ออกไปดอส');
264:            DisPlay:=1;  Choice:=1;
265:        end;

266: begin
267:  ReadText;
268:  if Max in [2..16] then
269:  begin
270:    Menu;
271:    repeat
272:      k:=0; Nosound;
273:      getimage(Xleft*9-7,Yup+1+(DisPlay-1)*28,
274:                  (Xleft+Width)*9+1,Yup+1+DisPlay*28,Data);
275:     repeat
276:      regs.AX:=$2C00; Intr($21,regs);
277:      if (Regs.DL in [0..49]) and (k=0) then
278:         begin k:=27;
279:          Putimage(Xleft*9-7,Yup+1+(DisPlay-1)*28,Data,4);
280:         end;
281:       if (Regs.DL in [50..99]) and (k=27) then
282:          begin k:=0;
283:           Putimage(Xleft*9-7,Yup+1+(DisPlay-1)*28,Data,0);
284:          end;
285:      until keypressed;
286:       Putimage(Xleft*9-7,Yup+1+(DisPlay-1)*28,Data,0);
287:       readchar;
288:      if Funckey in [#68,#72,#75,#77,#80] then
289:        begin
290:          case funckey of
291:           #68:begin
292:                RestoreCrtmode;  ExecDos;
293:                setgraphmode(Grmode);
294:                chdir(Path);     Menu;
295:               end;
296:           #72:if choice>1 then dec(Choice) else Choice:=Max;
297:           #80:if choice<max then inc(Choice) else Choice:=1;
298:           #75,#77:If Max>8 then
299:                     begin
300:                      if Xleft=XColumn1 then
301:                        begin
302:                         inc(Choice,MaxL);
303:                         if Choice>Max then dec(Choice,MaxL);
304:                        end
305:                        else dec(Choice,MaxL);
306:                     end;
307:          end; {CASE}
308:            DisPlay:=Choice;
309:           if (Funckey in [#72,#75,#77,#80]) and (Max>8) then
310:             begin
311:               if Choice<=MaxL then Xleft:=XColumn1
312:                else
313:                 begin
314:                  Xleft:=XColumn2; Display:=Choice-MaxL;
315:                 end;
316:             end;
317:         end;
318:       if (choice<=Max) and (ch=#13) then
319:         begin
320:           RestoreCrtmode;
321:           Runfile(Filename[choice]);
322:           setgraphmode(Grmode);
323:           chdir(Path);   Menu;
324:         end;
325:    until ch=#27;
326:   end;
327: end;

328: {==================== MAINMENU.PAS ==========================}
329: begin
330:  owner:='อาจหาญ สัตยารักษ์  ผู้ ปณ.34 อ.เมือง จ.ลำปาง 52000';
331:  getdir(0,Path);  Nosound;
332:  OpenGraph; Readfont;
333:  Mainmenu;
334:  chdir(Path);
335:  closegraph;
336: end.
337:
```

รูปที่ 7 ส่วนของการแสดงรายการเลือกของเมนูแบบเลื่อนแถบสว่างที่ปรับรูปแบบได้เอง

2. บรรทัดที่ 227 ตัวแปร choice เป็นตัวแปรแสดงลำดับรายการที่เลือก ส่วนตัวแปร Display จะเป็นรายการของเมนูแบบเลื่อนแถบสว่าง (ที่ใช้ 2 ตัวแปรเพราะว่ามี 2 คอลัมน์)

3. บรรทัดที่ 229-265 ส่วนการแสดงรายการเมนูบนจอภาพ ค่า Xleft, XColumn1, Xcolumn2 ใช้ในการหาตำแหน่งด้านซ้ายของเมนูแบบแถบสว่าง ส่วนด้านขวามืออาศัยตัวแปร Width ให้พิจารณาการใช้คำสั่งต่าง ๆ ด้วยตนเอง เพราะคิดว่าคงไม่ยากเกินกว่าการทำความเข้าใจ ตัวแปร MaxL จะเข้ามาช่วยในการปรับรายการเมนูให้เหมาะสม

4. บรรทัดที่ 270 แสดงรายการในเมนูบนจอภาพ

5. บรรทัดที่ 273-274 เก็บภาพรายการแรกของเมนูไว้ในตัวแปร Data (การที่ใช้ตัวแปร Data มาเก็บภาพเพราะตัวแปรนี้เราเลิกใช้ไปแล้ว จึงช่วยประหยัดหน่วยความจำได้ ห้ามลดค่าลงมากกว่านี้)

6. บรรทัดที่ 275-285 ส่วนการวนรอบแสดงรายการเมนูที่ติดต่อแบบ

กะพริบ

7. บรรทัดที่ 291-295 ออกไปดอสชั่วคราวเมื่อกดแป้น F10

8. บรรทัดที่ 296-306 การเลื่อนรายการในเมนูด้วยการกดแป้นลูกศรซ้ายขวาและขึ้นลง

9. บรรทัดที่ 308-317 ส่วนการตรวจรายการที่จะแสดงจากตัวแปร Display

10. บรรทัดที่ 318-324 การรันไฟล์ตามที่ระบุในแฟ้มเอกสาร menu.txt

คำสั่งนอกเหนือจากนี้ ผู้อ่านคงทำความเข้าใจได้ด้วยตนเอง ให้สังเกตว่าได้กำหนดคำสั่ง nosound ไว้ที่บรรทัดที่ 272 ทั้งนี้เพื่อป้องกันการออกไปรันไฟล์อื่นแล้วเกิดทำเสียค้างไว้

ก่อนจบขอให้ผู้อ่านย้อนไปดูบรรทัดที่ 1 ก่อน ซึ่งจะมีการกำหนดหน่วยความจำดังนี้

```
{SM 3000,0,4340}
```

เมื่อ 3000 คือ หน่วยความจำสแต็กที่โปรแกรมนี้ต้องการเมื่อมีการทำงาน ซึ่งตามความเป็นจริงนั้นต้องการใช้เพียง 2,300 ไบต์เท่านั้น ที่ผู้เขียนเพิ่มค่าให้อีก 700 ไบต์นั้นเพื่อให้รองรับไฟล์ *.bat ที่จะรันผ่านเมนูนั่นเอง (ถ้ากำหนดค่านี้น้อยไป เมื่อคอมพายล์เทอร์โบปาสคาลจะแจ้งข้อผิดพลาดให้ทราบ)

4340 คือ หน่วยความจำฮีพที่ส่วนการเข้ากราฟิกของไฟล์ *.obj ต้องการ ถ้ากำหนดน้อยเกินไป จะสามารถคอมพายล์ได้ แต่เมื่อรันไฟล์ *.EXE จะไม่สามารถเข้าสู่กราฟิกได้ โดยโปรแกรมจะแจ้งให้ทราบว่าเกิดความผิดที่ BGI Error: ซึ่งผู้อ่านบางท่านอาจเข้าใจผิดคิดว่าไม่ได้เรียนใช้คำสั่ง initgraph; ได้ ซึ่งที่จริงเกิดจากการกำหนดหน่วยความจำฮีพไม่พอ

ก่อนจาก

การทำงานจริงบนจอภาพนั้น ขอให้ผู้อ่านลองรันโปรแกรมดูจะเข้าใจมากขึ้น การทำงานจริง ๆ นั้น เราสามารถเปลี่ยนรายการเมนูและชื่อไฟล์ได้ตามความต้องการโดยใช้ภาษาไทยรหัสเกษตร

หวังว่าบทความนี้จะช่วยให้ผู้อ่านที่ต้องการเมนูเลือกไฟล์ทำงานที่เป็นภาษาไทยกราฟิก และสามารถรันไฟล์ได้มาใช้งานต่อไป

เมนู...เล่มล่า...

- การติดตั้งระบบ Config.sys และ Config.db 7
- จากดีเบสทรีไปเป็นดีเบสทรีพลัส 15
- การใช้ CATALOG 31
- การเขียนโปรแกรมด้วยดีเบสทรีพลัส 35
- เรื่องของตัวแปรและอะเรย์ 60
- ดีเบสกับเอ็มเอสดอส 70
- ความสัมพันธ์ระหว่างไฟล์ 77
- เวลาเป็นเงินเป็นทอง 87
- e^x log N และค่าทางตรีโกณมิติ 94
- ดีเบสกับการสร้างกราฟ 104
- ใช้ดีเบสทรีจัดเรียงคำภาษาไทย Thai Sort 122
- ดีเบสทรีพลัสกับภาษาแอสเซมบลี 131
- โปรแกรมจัดโครงสร้างโปรแกรม (Automatic Indent Lister: AIL version 1.01) 151

ฯลฯ

อิ่มเพียง 110 บาท

จาก

ไมโครคอมพิวเตอร์